Assignment 4 Machine learning

Yuhao Liao 201474244

SeyedJavad KhataeiPour 201793290

The aim of this assignment is designing a neural network that learns how to label the housing numbers.

Pre-processing steps.

The input is a 4D matrix which its last dimension indicated the number of images. We need to change the order of the input dimensions in a way that the number of images is the first and the number of channels is the last dimension. We used the transpose function from the Numpy library to rearrange the input matrix.

Moreover, to prepare the data, we changed the data from color to grayscale by using the inner product of the images with a vector. This will reduce the size of the images and increase the learning time. Also, images are normalized by dividing by 255. For the labes, firstly we change the label 10 to 0 as label ten is indicating the number 0. Secondly, we use a Keras function to make the labels categorical.

Network architecture:

In this assignment, we harvest the power of the convolutional neural networks to achieve very high accuracy. We do this by preprocessing the data, using three convolutional layers and one hidden layer and one output layer. Also, we use some normalization methods to cope with the overfitting problem. Table 1 shows a summary of the type of layers, activation function, regularization, etc. Also, a flowchart of the network structure is provided in Figure 1.

Table 1, Summary of the type of layers, activation function, regularization

Layer	Type	Parameter	Activation function	Regularizer
1	Conv2D	32 maps (3, 3)	Relu	
2	Conv2D	64 maps (5, 5)	Relu	
3	MaxPooling	Pool size (2,2)		
4	Conv2D	128 maps (3, 3)	Relu	
5	MaxPooling	Pool size (2,2)		
6	Dropout	Drop portion (0.2)		
7	Flatten			
8	Dense	Units (128)	Relu	L2 (0.0025)
9	Dense	Units (10)	Softmax	

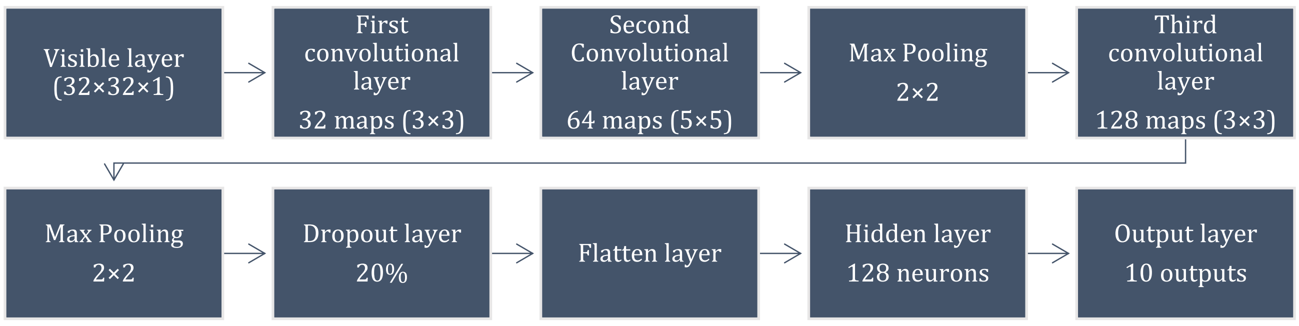

Figure 1, Summary of the Convolutional Neural Network Structure

Model selection justification:

One can use many different architectures for the model, yet not all of them provide a reasonable solution. In our design, we need to consider several points. First, we need to develop a model which can fit our training data with high accuracy. Second is the issue of overfitting. That is, if we do not use proper methods to control overfitting, our test accuracy will be unacceptable even though the training error is minimal. The third issue is the problem of the time and resources that we need to train our model. Using many layers might lead to a higher accuracy but for this assignment, our time and computational power are limited. For example, the model that Goodfellow et al., (2013) designed took six days to get trained. Here are our justifications for selecting the mentioned model considering the issues discussed above. [1]

Studying the Goodfellow et al., (2013) paper gave us an idea about a perfect model structure. They used "eight convolutional hidden layers, one locally connected hidden layer, and two densely connected hidden layers" [1]. However, due to our computational restrictions, we used three convolutional layers to detect different patterns from the images. Also, we used a fully connected layer after convolutional layers with 128 neurons to achieve a high accuracy. In the end, we place an output layer with 10 units as we have 10 possible outcomes.

To control overfitting, we used l2 regularizer with a penalties size of 0.0025 in the hidden layer. We tried l1 as well; however, the results were better with l2. Moreover, we used a dropout layer to ignore 20 % of the weights before feeding the data to the fully connected layer. Also, we used the max pooling technique to reduce the overfitting and to increase the training speed. We tried several models, and we released that although this technique makes the training process faster, it can reduce the overall accuracy. Therefore, it should be used with caution.

Finally, to control the training time, we convert the images to grayscale to reduce the input size. Each colored image has three channels. By changing them to grayscale, we reduce the number of channels to one and increase network training speed. Max Pooling and dropout techniques also helped to reduce training time. Moreover, we tried to keep the model as simple as possible by limiting the number of dense layers.

To conclude, we tried several different model structures with different parameters to find the best network. The presented model is the simplest model that can achieve an accuracy of above 92%.

How the model is trained:

After establishing the model, we need to compile the model and define a loss function, an optimizer, and a metric. For the loss function, with the help of the Keras documentation, we use the "categorical cross-entropy" function as we have ten different categories. We select Adam optimization algorithm as it is the efficient, default optimizer. For the metric we used accuracy which is the fraction of the images that are correctly classified.

When fitting the model, we need to decide the number of the epoch and the batch size. By trial and error, we found out that after... epochs this model stops improving so we need to stop after ten epochs. Also, the batch size we used is 100 as using all the data to train the model take a long time.

Results:

After ten epoch the test loss is 0.318 and the accuracy of the test data is 0.918. Table 2 shows the accuracy, loss and the number of epochs to achieve them for both the train and the test data. Figure 2 shows the predicted label and the accuracy for the first six test images. Figure 3 and Figure 4 show accuracy and loss function for each epoch respectively.

Table 2, Accuracy and loss for test and train data

Data type	Epochs	Accuracy	Loss
Train	10	0.9481	0.2805
Test	10	0.9184	0.3181

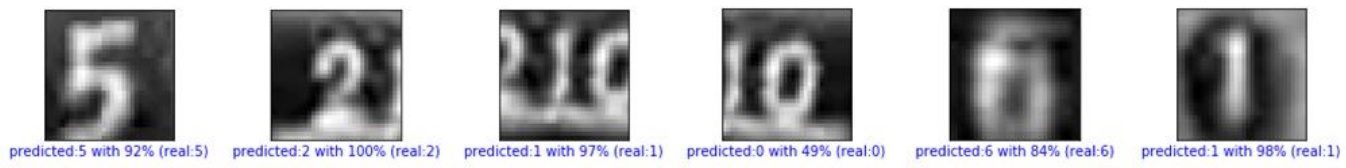
predicted:5 with 92% (real:5)
predicted:2 with 100% (real:2)
predicted:1 with 97% (real:1)
predicted:0 with 49% (real:0)
predicted:6 with 84% (real:6)
predicted:1 with 98% (real:1)

Figure 2, Six test images, with predicted label, the accuracy of the prediction and the real label

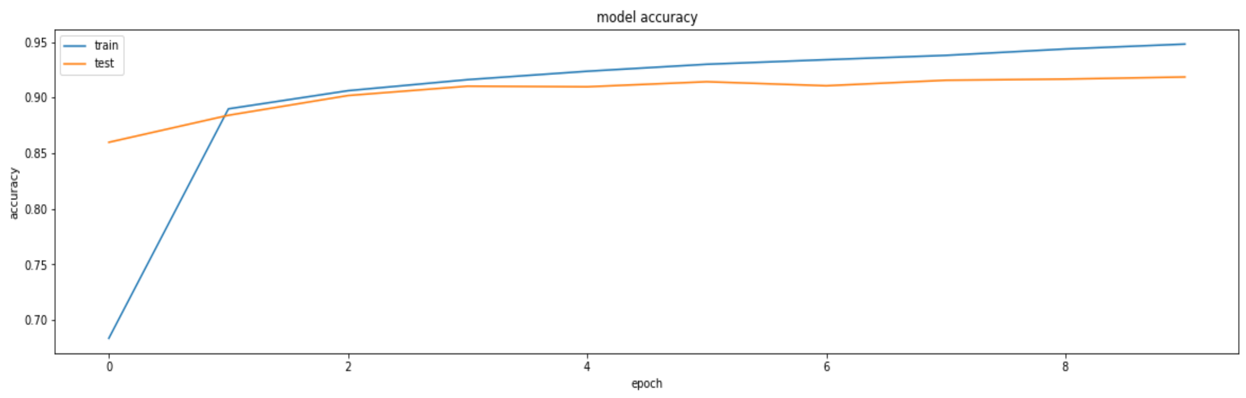

Figure 3, Accuracy vs epoch for test and train data

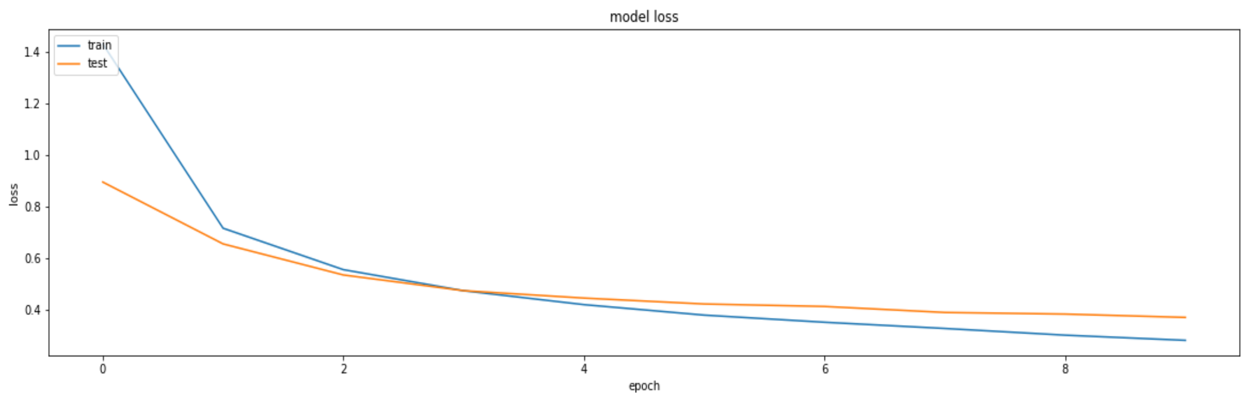

Figure 4, loss vs epoch for test and train data

References:

1- Goodfellow, I. J., Bulatov, Y., Ibarz, J., Arnoud, S., & Shet, V. (2013). Multi-digit number recognition from street view imagery using deep convolutional neural networks. *arXiv preprint arXiv:1312.6082*.